Development Plan
RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept. 10, 2025	Mike Chen	Proof-of-Concept Plan and Expected Technology
Sept. 13, 2025	Jianqing Liu	Team Member Roles and Communication Plan
Sept. 13, 2025	Jianqing Liu	Expected Technology and Coding Standard
Sept. 20, 2025	Jianqing Liu	Team Meeting Plan, Workflow Plan and Scheduling
Sept. 21, 2025	Jianqing Liu	Misc changes and expedited workflow plan

This document outlines the development plan for RoCam, a high performance vision-guided rocket tracker.

1 Confidential Information?

No confidential information to protect.

2 IP to Protect

No IP to protect.

3 Copyright License

This project adopts the MIT license, which is available at this link.

4 Team Meeting Plan

The team will meet from 7:00–8:00 PM every Tuesday and Saturday. Meetings will be held online using Discord. If an in-person session is required for hardware testing, a separate time and location will be scheduled in advance of each event.

Each team meeting will be structured as follows:

1. The agenda for each meeting will be posted as a GitHub issue ahead of time.
2. Each team member will share task updates (progress, difficulties).
3. Discuss and distribute new tasks to be worked on.
4. (Saturdays only) Draft an email for the supervisor summarizing weekly progress and any questions.

Communication with the supervisor will include a weekly update email and optional online or in-person meetings if either party requests them.

5 Team Communication Plan

- **Discord**: General team communication, informal discussions, quick updates, and meetings.
- **GitHub**: Code-related discussions, project management, and meeting notes.
- **Zoom**: Meetings with supervisor.
- **Email**: Weekly progress updates to supervisor.

6 Team Member Roles

- **Project Manager**: Oversees project timeline, coordinates tasks between team members, manages deliverables and deadlines, and serves as primary point of contact with supervisor and stakeholders.
- **Meeting Chair**: Leads team meetings, prepares agendas, ensures discussions stay on track, facilitates decision-making, and manages meeting time effectively.
- **Notetaker**: Records meeting minutes, tracks action items and decisions, maintains documentation of team discussions, and distributes meeting summaries to all members.

- **Quality Assurance**: Reviews code, documentation, and deliverables for quality and consistency, conducts testing and validation, ensures adherence to coding standards and project requirements, and manages the review process for all team outputs.

7 Workflow Plan

7.1 Normal Features

1. Planning
   (a) Create an issue in GitHub Projects under "Backlog" using an appropriate template (bug or enhancement), and assign it to the correct subproject.
   (b) Backlog issues will be discussed during meetings to refine scope and requirements.
   (c) If the issue is approved for development, assign an owner and a deadline, then move it into "Todo".
2. Developing
   (a) The assignee will work on the task in a new branch: [main-contributor-name]/[feature-name].
   (b) Move the issue into "In Progress".
   (c) Create a pull request once the code is ready for review. The pull request should reference the original issue.
   (d) Request a review from at least one team member and ping them on Discord.
3. Reviewing
   (a) Move the issue into "In Review".
   (b) The reviewer may comment or commit directly to the feature branch.
   (c) The reviewer approves and merges the pull request.
   (d) Delete the feature branch after the pull request is merged.
   (e) Move the issue into "Done".

7.2 Minor Changes (Expedited Workflow)

For small, low-risk tasks (e.g., fixing typos or minor UI adjustments), the normal workflow may be skipped to reduce overhead. Creating an issue on the GitHub Projects board is not required. Instead:

1. Open a pull request against the main branch with "minor" label attached.
2. Obtain at least one reviewer approval before merging.
3. Delete the feature branch after the pull request is merged.

8 Project Decomposition and Scheduling

This project is decomposed into the following subprojects:

- **Web App**: responsible for remote management.
- **CV Pipeline**: responsible for locating the rocket.
- **Motion Control**: responsible for controlling gimbal movement.

All code for the subprojects, along with documentation, is centralized in a single monorepo here. GitHub Projects is used for project management and can be accessed here.

While development will be broken down into smaller features with individual deadlines, the overall project will follow the major deadlines below.

Table 2: Major Deliverables

Date	Deliverable	Files
Sept. 22, 2025	Problem Statement, POC Plan, Development Plan	Problem Statement POC and Development Plan
Oct. 6, 2025	Req. Doc. and Hazard Analysis Revision 0	Req. Doc. Hazard Analysis
Oct. 27, 2025	V&V Plan Revision 0	V&V Plan
Nov. 10, 2025	Design Document Revision -1	Design Document
Nov. 17-28, 2025	Proof of Concept Demonstration	
Jan. 19, 2026	Design Document Revision 0	Design Document
Feb. 2-13, 2026	Revision 0 Demonstration	
Mar. 9, 2026	V&V Report and Extras Revision 0	V&V Report Extras
Mar. 23-29, 2026	Final Demonstration (Revision 1)	
TBD	EXPO Demonstration	
Apr. 6, 2026	Final Documentation (Revision 1)	

9 Proof-of-Concept Demonstration Plan

The following are the planned steps of the POC:

1. Acquire an initial image from a camera with a stationary target.
2. Activate the system's tracking mode. It will segment and detect multiple moving objects in the image.
   - Segmentation and moving-object detection will use computer vision techniques.
   - The computer vision model will be deployed on a Jetson Nano.
3. The user selects a stationary or moving object as the target.
4. As the target moves, the system keeps it centered in the image for smooth tracking.
   - The system will handle occlusion and temporary loss of the target.
   - The user can manually reselect the target if needed.
   - Real-time camera control will be implemented using an STM32 microcontroller.

The following is a list of primary risks to consider for the POC:

1. The computer vision system may not process images at a sufficient frame rate.
   - If this occurs, we will optimize the existing model, consider using a more powerful board, lower the frame rate, or use a traditional algorithmic approach that detects motion via pixel-wise image comparison.
   - We also reserve the option to use models trained for a specific set of objects to increase throughput.
2. The STM32 may not deliver real-time control to the camera.
   - If this occurs, we will consider using a more powerful microcontroller, or a different control technique.
3. Integration of the frontend, backend, computer vision model, and microcontroller may be more difficult than expected.
   - If this occurs, we will consider using a more powerful integrated single-board computer, deploying via cloud computation, or redesigning our control flow to simplify integration.

Other smaller risks to consider:

1. UI usability issues: The user interface may not be intuitive or easy to use, leading to user frustration or errors.
    - Potential solution: Conduct user testing and gather feedback to improve the interface design.

10 Expected Technology

- Motion Control
    - STM32 Microcontroller
    - Language: Rust
    - Framework: embassy-rs
    - Formatting: rustfmt
    - Linter: rust-clippy
    - Unit Testing: Rust built-in
    - Code Coverage: grcov
- Computer Vision
    - NVIDIA Jetson
    - NVIDIA JetPack
    - Language: Python
    - Libraries: OpenCV, NumPy, Matplotlib, Torch
    - Open Source Models: Ultralytics YOLO, SAM, various ViTs
    - Formatting: Black
    - Linter: pylint and ruff
    - Unit Testing: pytest
    - Code Coverage: coverage
    - Containerized using Docker
- Web App
    - Web Server: Flask
    - Language: TypeScript
    - Framework: React
    - Formatting: prettier
    - Linter: eslint
    - End-to-end Testing: Cypress
- All of the above will use GitHub Actions for CI.
- Development Tools
    - VS Code
    - PyCharm
    - Git
    - GitHub

11 Coding Standard

- Rust: The Power of 10 Rules
- Python: PEP 8
- TypeScript: typescript-eslint

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. Why is it important to create a development plan prior to starting the project?
2. In your opinion, what are the advantages and disadvantages of using CI/CD?
3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Appendix — Team Charter

External Goals

Attendance

Expectations

Acceptable Excuse

In Case of Emergency

Accountability and Teamwork

Quality

Attitude

Stay on Track

Team Building

Decision Making